Sage for Number Theorists

http://www.sagemath.org

William Stein[1]

[1]Department of Mathematics
University of Washington, Seattle

Bristol, 2007-11-14

Outline

A Quote from Neil Sloane from Last Week

Neil Sloane

```
From: N. J. A. Sloane <njas@research.att.com>
Date: 8 Nov 2007 06:28
Subject: Re: dumb question about installing pari-gp with fink

I would like to thank everyone who responded to
my question about installing PARI on an iMAC.

The consensus was that it would be simplest to install sage,
which includes PARI and many other things.

I tried this and it worked!

Thanks!

Neil

(It is such a shock when things actually work!!)
```

OK, so what is this “Sage” that N.J.A. Sloane is raving about?

Sage: Mission Statement

Provide a single open source high-quality **viable alternative** to **Magma, Mathematica, Maple** and **MATLAB**.

Do not reinvent the wheel but **reuse** as much **existing building blocks** as possible and make sure the result is **rigorously tested**, **easy to modify** by the end user and **very well documented**.

Also create a **helpful environment** and community to get help (mailinglists, irc-channel, meetings, coding sprints). This helps not only development but day-to-day research.

The Inquirer: Mathematics rediscovers the scientific method

This is from British magazine **The Inquirer** (`http://www.theinquirer.net`)!!

Mathematics rediscovers the scientific method

In open source software; By Egan Orion: Wednesday, 24 October 2007

IN AN OPINION published by the American Mathematical Society, David Joyner and William Stein argue that the use of closed, proprietary mathematical software is fundamentally incompatible with the standards of mathematical proof.

They note that at least one published article on mathematical theory has relied upon the use of proprietary software to deduce various mathematical facts. They see a disconnect in this practice for the checkability of mathematical reasonings, up to and including the proofs of new mathematical theorems. They write:

> *"Increasingly, proprietary software and the algorithms used are an essential part of mathematical proofs. To quote J. Neubuser,* **'with this situation two of the most basic rules of conduct in mathematics are violated: In mathematics information is passed on free of charge and everything is laid open for checking.'***"*

It's somewhat **astonishing to contemplate** that mathematicians even **got so lazy as to trust the inner workings of closed software algorithms**, but **perhaps all it proves is that they're human too, after all**.

What is Sage?

Sage is a very large mathematics software package developed by a worldwide community of developers. Sage is:

1. a **distribution** of the best free, open-source mathematics software available (Sage 2.8.12 ships over 50 third-party packages) that is easy to compile or install from binaries.
2. a **huge new library**, which uniformly covers the widest area of **functionality**, including several new implementations not yet found elsewhere.
3. **interfaces** to almost all existing mathematics software packages (including Magma, PARI, Gap, Mathematica, Maple, etc.)

Welcome to Sage:

```
| SAGE Version 2.8.12, Release Date: 2007-11-06
| Type notebook() for the GUI, and license() for information.
```

```
sage: EllipticCurve('389a')
Elliptic Curve defined by
          y^2 + y = x^3 + x^2 - 2*x
over Rational Field

sage: factor(2^129 + 1)
3^2 * 1033 * 1591582393 * 29320031007403 * 15686603697451
```

Python & Cython: The Languages of Sage

http://www.python.org and http://www.cython.org

Python

- easy for you to define **your own data types** and methods on it (bitstreams, ciphers, rings, whatever).
- very clean language that results in **easy to read code**.
- easy to learn (several good books, some free).
- a **huge number of libraries**: statistics, networking, databases, bioinformatic, physics, video games, 3d graphics, numerical computation (scipy), and serious "pure" mathematics (via Sage)

Cython

- a Python compiler (and more)
- allows to mix C and Python which is crucial for fast execution speed
- makes easy to **use existing C/C++ libraries** from Python.
- project started by me as a fork of Pyrex
- Robert Bradshaw (my first Ph.D. student) is the project leader

Outline

Some Capabilities

see `http://wiki.sagemath.org/cando`

Number Theory (mwrank, Simon, Lcalc GP/PARI, NTL) **Mordell-Weil groups**, very fast **quadratic sieve**, **modular symbols for general** weight, character, $\Gamma_1(N)$, and $\Gamma_H(N)$, **Kedlaya's point-counting algorithm** (Harvey's fast variant), Coleman integration; **Coleman integration and p-adic heights**.

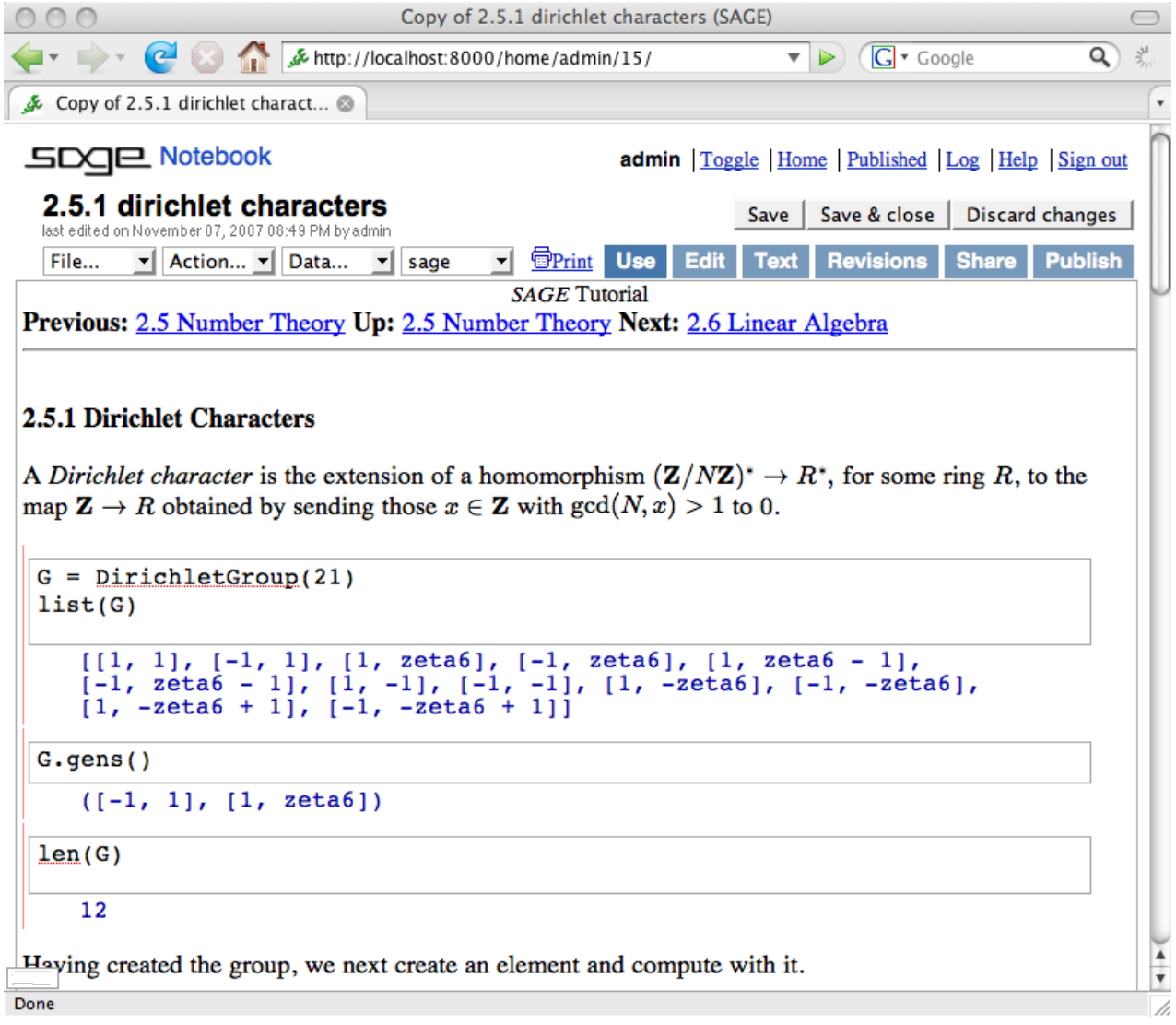
Elliptic Curves All the standard stuff, plus **complex and p-adic L-functions**, and fast computation of **p-adic heights and regulators for** $p < 100000$ with new algorithm.

p-adic Numbers Highly optimized support for arithmetic with a wide range of different models of **p-adic arithmetic**.

And much much more...

Bonus: A Powerful Web-based Graphical User Interface

public notebooks available at `http://www.sagenb.org`



- graphical user interface
- plotting
- LaTeX typesetting
- remote access
- worksheet sharing
- interface to 3rd party systems, e.g. Magma

Outline

Integer Factorization

factor uses PARI

```
sage: time factor(next_prime(2^40) * next_prime(2^300),verbose=0)
1099511627791 *
203703597633448608626844568840937816105146839366593625063614044935438129\
9763336706183397533
CPU time: 3.77 s,  Wall time: 3.82 s
```

ecm uses GMP-ECM by Paul Zimmermann et al.

```
sage: time ecm.factor(next_prime(2^40) * next_prime(2^300))
[1099511627791, 203703597633448608626844568840937816105146839366593625063614044935438129
CPU time: 0.19 s,  Wall time: 0.62 s
```

qsieve uses Bill Hart's quadratic sieve implementation

```
sage: v,t = qsieve(next_prime(2^90)*next_prime(2^91),time=True)
sage: print v, t[:4]
[1237940039285380274899124357, 247588007857076054979824850 7] 4.00
```

DistributedFactor combines ECM, qsieve and trial division; written by Yi Qiang and Robert Bradshaw.

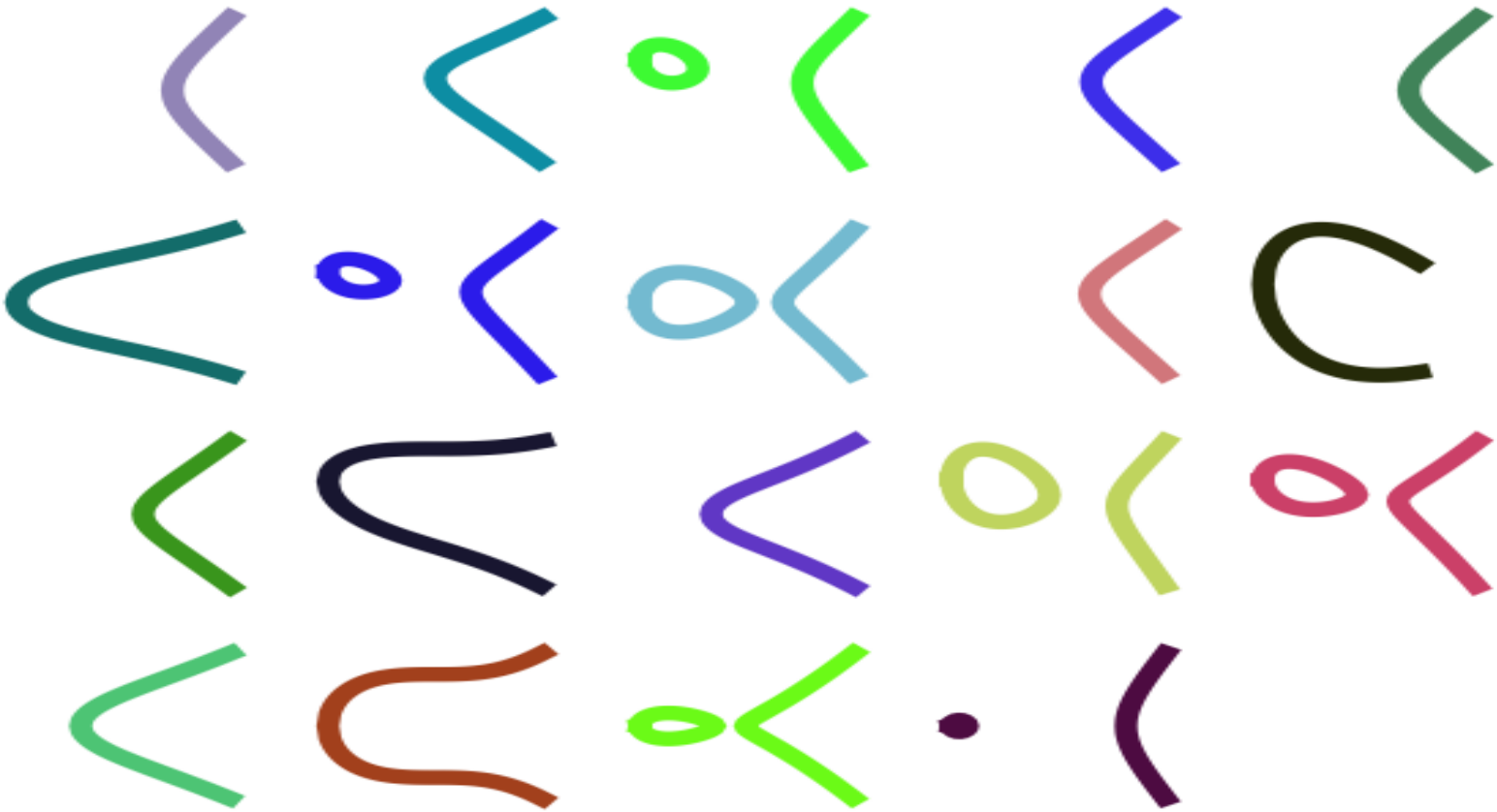
LLL Lattice Basis Reduction

Sage includes several LLL implementations, including PARI's, NTL's, and **Damien Stehle's fpLLL** (currently the world's best).

```
sage: from sage.libs.fplll.fplll import gen_ntrulike
sage: A = gen_ntrulike(200,130,35)
sage: time B = A.LLL()   # uses fpLLL by Damien Stehle
CPU time: 14.91 s,  Wall time: 15.06
```

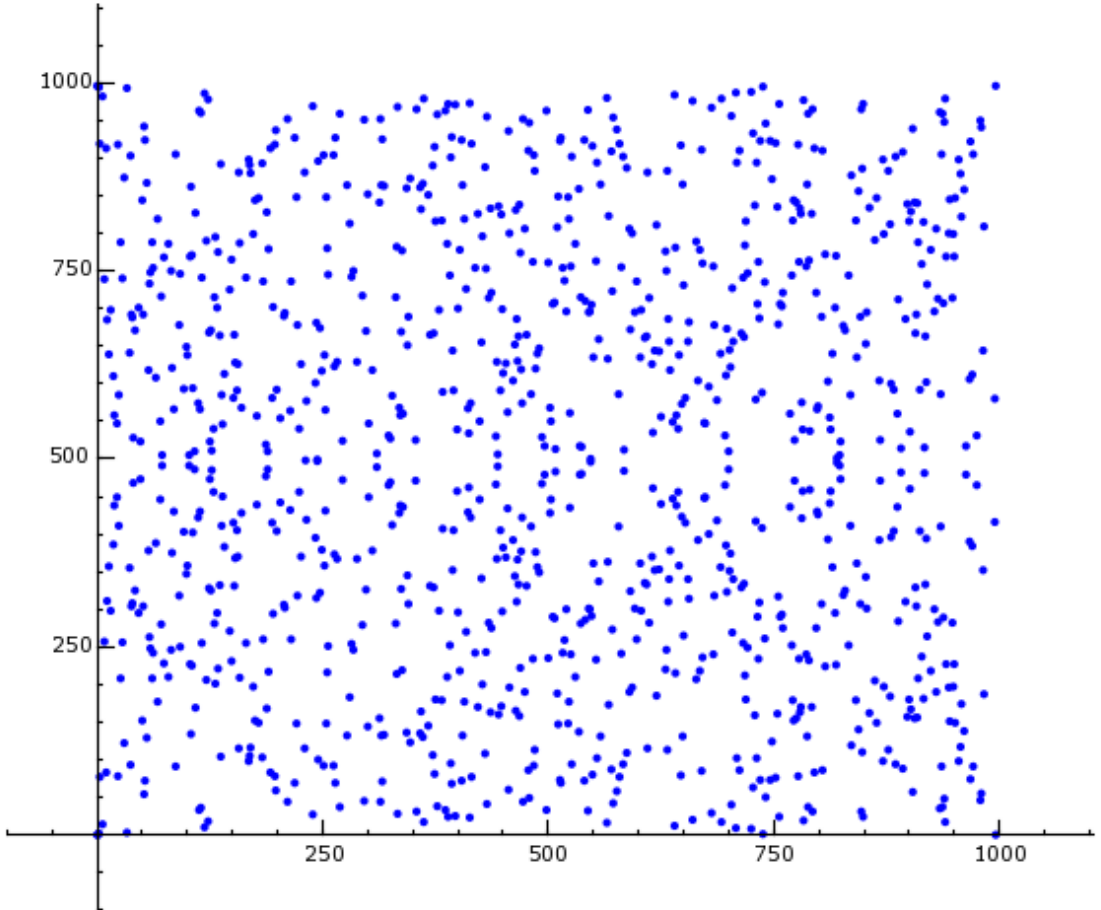
Elliptic Curves for Kids

```
X = []
for E in cremona_optimal_curves([11..37]):
    X.append(plot(E, thickness=10,
            rgbcolor=(random(),random(),random())))
show(graphics_array(X, 4,5), axes=False)
```



Plot elliptic curves over finite fields:

```
sage: e = EllipticCurve("37a") # Cremona Label
sage: E = e.change_ring(GF(997))
sage: show(E.plot())
```



Sage includes a fast SEA implementation.

```
sage: k = GF(next_prime(10^20))
sage: E = EllipticCurve(k,
...       [k.random_element(),k.random_element()])
sage: time E.cardinality()
CPU times: user 0.00 s, sys: 0.02 s, total: 0.02 s
Wall time: 0.56
99999999985979523788
```

Elliptic Curve and p-adic BSD / Iwasawa Theory

```
sage: E = EllipticCurve('389a')

sage: L = E.padic_lseries(5)
sage: L.series(3)
O(5^5) + O(5^2)*T + (4 + 4*5 + O(5^2))*T^2
   + (2 + 4*5 + O(5^2))*T^3 + (3 + O(5^2))*T^4 + O(T^5)

sage: E.padic_regulator(5,10)
5^2 + 2*5^3 + 2*5^4 + 4*5^5 + 3*5^6 + 4*5^7
    + 3*5^8 + 5^9 + O(5^11)

sage: time E.padic_regulator(997,10)
CPU times: user 0.46 s, sys: 0.01 s, total: 0.47 s
740*997^2 + 916*997^3 + 472*997^4 + 325*997^5 + 697*997^6
  + 642*997^7 + 68*997^8 + 860*997^9 + 884*997^10 + O(997^11)
```

Modular Symbols

Sage has the most general implementation of modular symbols available, though much remains to be done:

```
sage: M = ModularSymbols(GammaH(13,[3]), weight=4)
sage: print M.basis()
([X^2,(0,1)], [X^2,(0,7)], [X^2,(2,5)], [X^2,(2,8)], ...
sage: factor(charpoly(M.T(2)))
(x - 7) * (x + 7) * (x - 9)^2 * (x + 5)^2
        * (x^2 - x - 4)^2 * (x^2 + 9)^2
sage: dimension(M.cuspidal_subspace())
12
sage: [a.dimension() for a in M.decomposition(3)]
[1, 1, 2, 2, 4, 4]
```

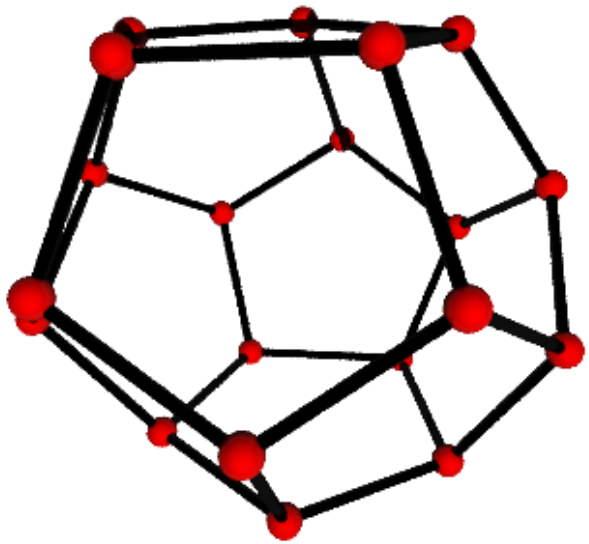
Will soon also have John Cremona's specialized but **insanely fast** implementation for weight 2 trivial character:

```
sage: import sage.libs.cremona.homspace as jc
sage: time M = jc.ModularSymbolsSpace(20014)
CPU times: user 1.72 s, sys: 0.13 s, total: 1.84 s
sage: time t = M.hecke_matrix(2, convert=False)
CPU times: user 0.79 s, sys: 0.53 s, total: 1.33 s
```

Graph Theory: sometimes relevant to number theory

- builds on NetworkX (Los Alamos's Python graph library)
- graph isomorphism testing – Robert Miller's new implementation
- databases
- 2d and 3d visualization

```
sage: D = graphs.DodecahedralGraph()
sage: D.show3d()
```



```
sage: E = D.copy()
sage: gamma = SymmetricGroup(20).random_element()
sage: E.relabel(gamma)
sage: D.is_isomorphic(E)
True
sage: D.radius()
5
```

Polynomial Arithmetic

- Sage mostly currently uses NTL by default
- Soon use FLINT extensively – world's fastest univariate polynomial arithmetic for essentially every bit size and degree (Bill Hart's and David Harvey).

```
sage: from sage.libs.flint.fmpz_poly import Fmpz_poly
sage: deg = 31; c=64
sage: f=Fmpz_poly([ZZ.random_element(2^c) for _ in [1..deg+1]])
sage: g=Fmpz_poly([ZZ.random_element(2^c) for _ in [1..deg+1]])
sage: time for _ in xrange(10^5): w = f*g
CPU time: 1.55 s,  Wall time: 1.67 s
```

- PARI takes 9.85 seconds to do the exact same computation.
- Sage wrapping NTL takes 9.24 seconds
- Magma takes 4.68 seconds

Commutative Algebra

- Very fast basic arithmetic for multivariate polynomials over $\mathbb{F}_q$.
- Fast Groebner basis computation and ideal operations

The Fateman benchmark:

```
sage: P.<x,y,z> = PolynomialRing(GF(32003),3)
sage: p = (x + y + z + 1)^20 # the Fateman fastmult benchmark
sage: q = p + 1
sage: t = cputime()
sage: r = p*q
sage: cputime(t)
0.13
```

Magma takes 0.360 seconds to do the same calculation (both on linux 64-bit)

Outline

History of Sage

1997–1999 (**Berkeley**) **HECKE** – C++ (modular forms)

1999–2005 (**Berkeley**, **Harvard**) I wrote over 25,000 lines of **Magma** code.

But the languages, devel models, and quality standards of Magma, Mathematica, and Maple are **old-fashioned and painful**.

I need to be able to **see inside and change anything** in my software in order to be the best in the world at my research.

For me, Magma is a bad long-term investment.

Feb 2005 I released **SAGE-0.1**

Feb 2006 **UCSD SAGE Days 1** workshop – SAGE 1.0.

October 2006 **U Washington SAGE Days 2** workshop.

March 2007 **UCLA SAGE Days 3** workshop.

May 2007 First Sage NSF grant.

June 2007 **U Washington SAGE Days 4** workshop.

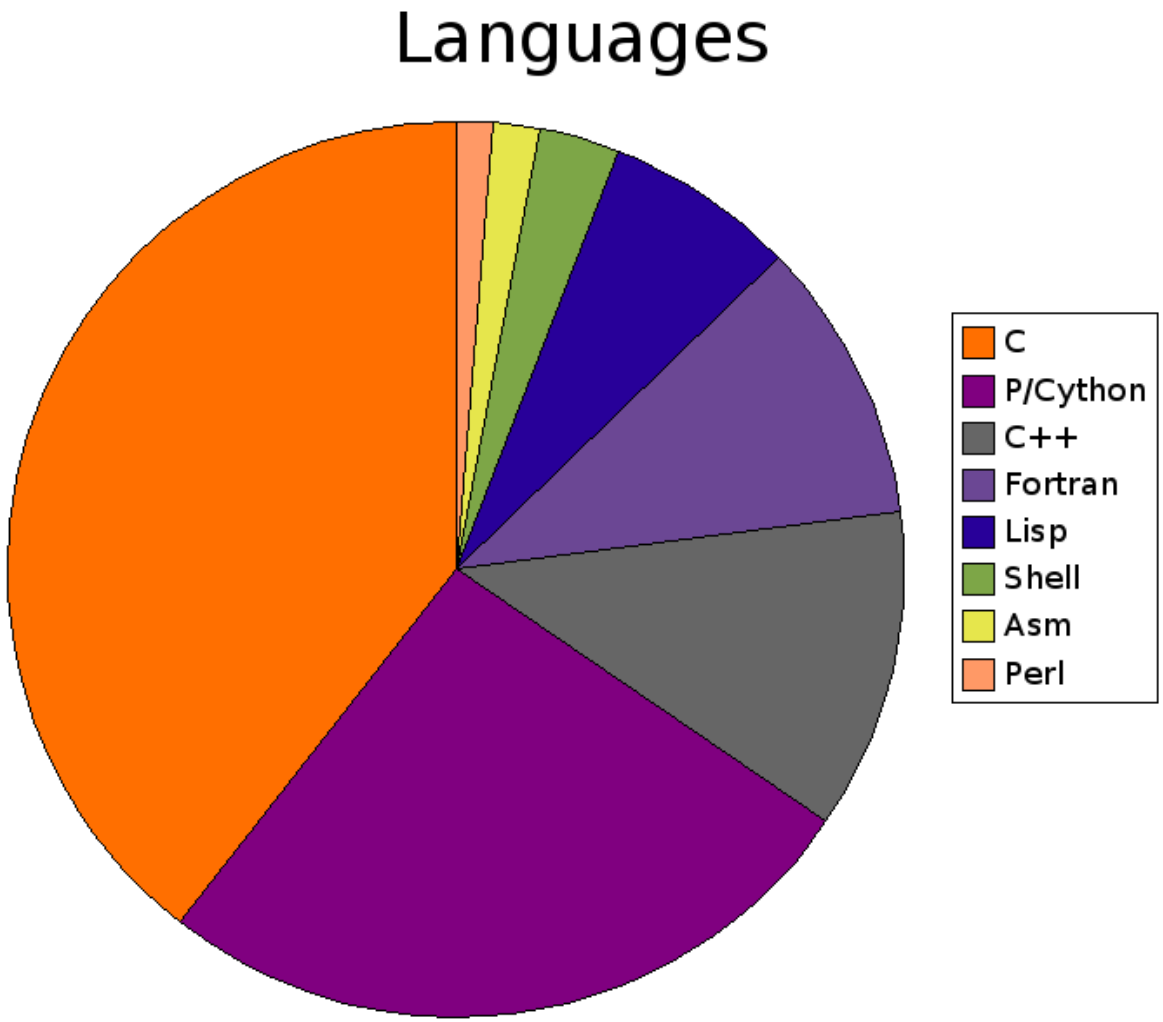
October 2007 **Clay Math Institute SAGE Days 5** workshop.

Now **Heilbronn Institute SAGE Days 6**

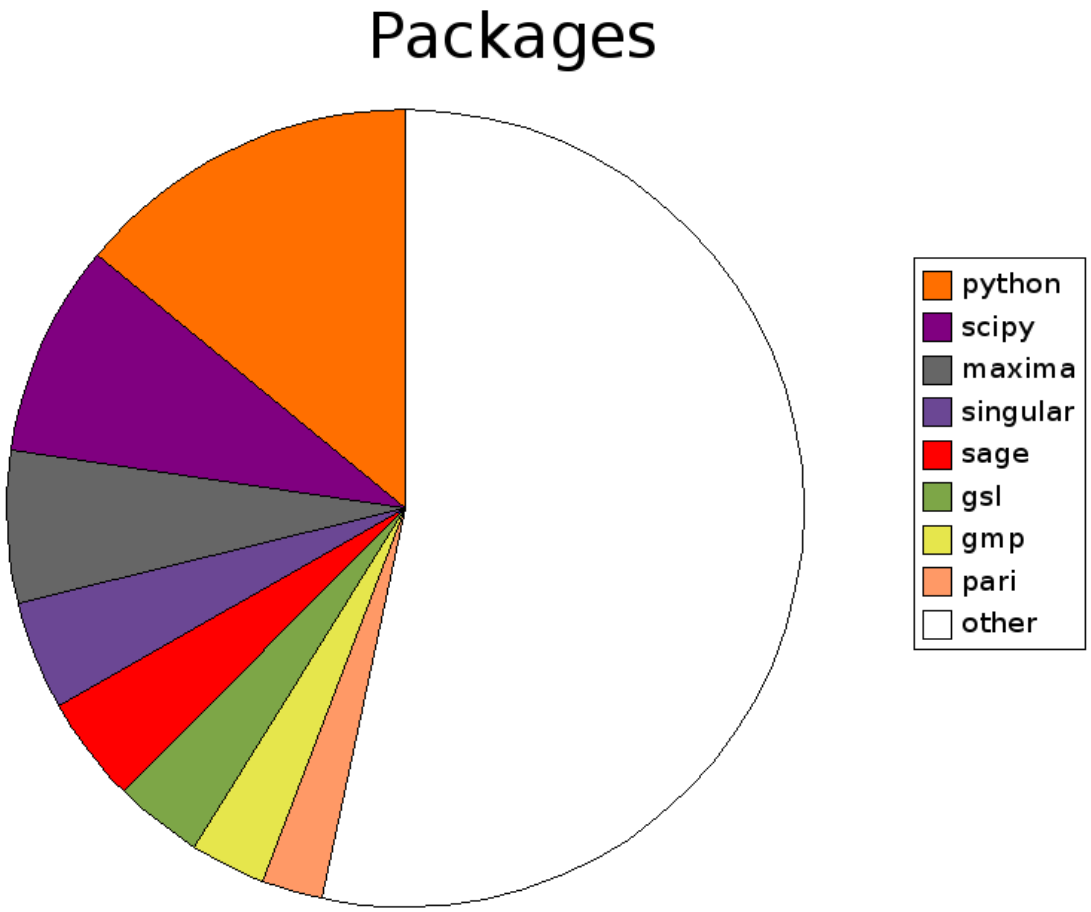
Now **SAGE-2.8.12**; well over **100 contributors to SAGE**.

Do not reinvent the wheel:

Arithmetic	**GMP, MPFR, Givaro**
Commutative Algebra	**SINGULAR** (libSINGULAR)
Linear Algebra	**LinBox, M4RI, IML, fpLLL**
Cryptosystems	**GnuTLS, PyCrypto**
Integer Factorization	**FLINT-QS, ECM**
Group Theory	**GAP**
Combinatorics	**Symmetrica**
Graph Theory	**NetworkX**
Number Theory	**PARI, NTL, FLINT, mwrank**
Numerical Computation	**GSL, Numpy, Scipy**
Calculus, Symbolic Comp.	**Maxima, Sympy**
Lattice Polytopes	**PALP**
User Interface	Sage Notebook, **jsmath, Moin wiki, IPython**
Graphics	**Matplotlib, Tachyon, libgd, Java3d**
Networking	**Twisted**
Databases	**ZODB, SQLite**, Python Pickles
Programming Language	**Python, Cython** (compiled)



A total of nearly 5 million lines of source code (several hundred person-years).


Packages
python
scipy
maxima
singular
sage
gsl
gmp
pari
other

SAGE: Lots of new code

Unique lines of code and docstrings:

```
$ cat *.py */*.py ... */*/*/*.pxd |sort |uniq |wc -l
189082
```

Unique input lines of autotested examples:

```
$ cat *.py */*.py ... */*/*/*.pxd | grep "sage:"
              | sort |uniq |wc -l
26711
```

Doctesting coverage:

```
$ sage -coverage .
...
Overall weighted coverage score:  34.3%
Total number of functions:  17424
```

MANY Upcoming Workshops

- **Jan 5–9, 2008:** Sage Days $6\frac{1}{4}$, AMS meeting in San Diego (booth, sprints)
- **Feb 5–9, 2008:** Sage Days 7, IPAM (confirmed and funded!)
- **Feb 29–March 4, 2008:** Sage Days 8, UT Austin – **Quadratic Forms**
- **June 2008:** Sage Days 9 in Seattle (tentative)
- **August 2008:** Sage Days 10 in Vancouver at SFU (tentative) – organized by Nils Bruin
- And many more...

Sage Development: Rapid but High Quality

1. All enhancement proposals, bug reports and tasks are available on `http://trac.sagemath.org`.
2. All discussions happen in the open on **public mailing lists** and on a public chat channel.
3. **One stable release per week (!)** on average: release often, release early.
4. Code is **refereed**.
5. Several different **release managers**.

Outline

Shortcomings of Sage

1. There are currently **probably less than a thousand** serious users of Sage; our goal is to have ten thousand serious users by 2009.

2. Sage is **not** of high enough quality yet.

3. Sage is sometimes **much slower** than Magma, Mathematica, etc.

Advantages of Sage

1. only general purpose mathematics software that **uses a mainstream programing language**.
2. genuinely allows you to use Maple, Mathematica, Magma, etc., all **together**.
3. **more functionality out of the box** than any other open source mathematics software.
4. Sage has a **large supportive and active community**: active mailing lists have well over 200 subscribers – "very positive vibe"
5. Sage development is **done in the open**.
6. Sage is sometimes **much faster** than Magma, Mathematica, etc.

A Quote from John Voight's MIT Talk Last Month

John Voight

"Having seemingly eliminated every alternative, *we turn to SAGE.* SAGE includes Pari, so it has number field arithmetic. It uses Python, which is a very friendly modern (object-oriented) programming language. It is free. It incorporates Cython, which easily allows one to write optimized C code for repeated tasks.

Despite being a relatively new system (so some functionality is limited), since it is open source it is easy to contribute yourself. For example, Carl Witty recently wrote a package for fast real root isolation. So even though one must think about issues like how best to coerce between a C int, a Python integer, and a SAGE Integer, there is a very active development community willing to help!

It has the further advantage that there is a package for **distributed computing called DSage**..."

Questions?

Thank You!